Network Working Group
Request for Comments: 4252
Category: Standards Track

T. Ylonen
SSH Communications Security Corp
C. Lonvick, Ed.
Cisco Systems, Inc.

January 2006

Протокол аутентификации SSH

The Secure Shell (SSH) Authentication Protocol

Статус документа

В этом документе содержится спецификация протокола, предложенного сообществу Internet. Документ служит приглашением к дискуссии в целях развития и совершенствования протокола. Текущее состояние стандартизации протокола вы можете узнать из документа Internet Official Protocol Standards (STD 1). Документ может распространяться без ограничений.

Авторские права

Тезисы

Протокол SSH[1] используется для организации безопасного входа в удаленную систему (login) и организации иных безопасных служб через сети, не обеспечивающие защиты. В данном документе описан протокол аутентификации SSH, а также методы аутентификации клиентов на основе открытых ключей, паролей и хостов. Дополнительные методы аутентификации рассматриваются в отдельных документах. Протокол аутентификации работает на основе транспортного протокола SSH и обеспечивает создание надежного туннеля для протокола соединений SSH.

Оглавление

1. Введение

Протокол аутентификации SSH представляет собой протокол общего назначения, используемый для проверки подлинности пользователей. Этот протокол предназначен для работы поверх транспортного протокола SSH [SSH-TRANS]. Протокол аутентификации предполагает, что нижележащие протоколы обеспечивают целостность и конфиденциальность данных.

Данный документ следует читать только после прочтения описания архитектуры SSH, содержащегося в [SSH-ARCH]. В документе используется терминология и нотация, описанные в посвященном архитектуре протокола документе без каких-либо ссылок или пояснений.

Имя службы ('service name') для протокола аутентификации - "ssh-userauth".

На начальном этапе работы протокол получает идентификатор сессии от протокола нижележащего уровня (хэш H, полученный от первого обмена ключами). Идентификатор сессии служит уникальным признаком данной сессии и может применяться в качестве «подписи» для обозначения владельца закрытого ключа (private key). Протоколу также требуется знать, обеспечивает ли нижележащий уровень защиту конфиденциальности данных.

[1] Secure Shell

2. Разработчики

Основными разработчиками этого комплекта документов являются: Tatu Ylonen, Tero Kivinen, Timo J. Rinne, Sami Lehtinen (все из SSH Communications Security Corp) и Markku-Juhani O. Saarinen (университет Jyvaskyla). Darren Moffat был редактором этого комплекта документов и внес важный вклад в работу.

За годы подготовки этого документа множество людей внесло свой вклад. В их число входят: Mats Andersson, Ben Harris, Bill Sommerfeld, Brent McClure, Niels Moller, Damien Miller, Derek Fawcus, Frank Cusack, Heikki Nousiainen, Jakob Schlyter, Jeff Van Dyke, Jeffrey Altman, Jeffrey Hutzelman, Jon Bright, Joseph Galbraith, Ken Hornstein, Markus Friedl, Martin Forssen, Nicolas Williams, Niels Provos, Perry Metzger, Peter Gutmann, Simon Josefsson, Simon Tatham, Wei Dai, Denis Bider, der Mouse и Tadayoshi Kohno. Указанные в списке люди могли не участвовать в написании данного документа, но они внесли свой вклад в его подготовку.

3. Используемые в документе соглашения

Во всех документах, связанных с протоколом SSH, следует использовать ключевые слова: **необходимо** (MUST), **недопустимо** (MUST NOT), **требуется** (REQUIRED), **нужно** (SHALL), **не нужно** (SHALL NOT), **следует** (SHOULD), **не следует** (SHOULD NOT), **рекомендуется** (RECOMMENDED), **возможно** (MAY), **необязательно** (OPTIONAL) для описания уровня требования. Интерпретация этих слов описана в [RFC2119].

Ключевые слова **приватное использование** (PRIVATE USE), **иерархическое выделение** (HIERARCHICAL ALLOCATION), **выделение в соответствии с порядком запросов** (FIRST COME FIRST SERVED), **экспертное рассмотрение** (EXPERT REVIEW), **требуется спецификация** (SPECIFICATION REQUIRED), **одобрение IESG** (IESG APPROVAL), **согласование с IETF** (IETF CONSENSUS), **стандартизация** (STANDARDS ACTION) в данном документе при их использовании в контексте распределения пространства имен интерпретируются в соответствии с [RFC2434].

В данном наборе документов определяются поля протокола и возможные значения этих полей. Поля будут определяться вместе с протокольными сообщениями. Например, поле SSH_MSG_CHANNEL_DATA определяется следующим образом.

```
byte      SSH_MSG_CHANNEL_DATA
uint32    recipient channel (канал получателя)
string    data (данные)
```

В данном документе поля протокола будут указываться в одинарных кавычках, а значения полей – в двойных. В приведенном выше примере поле 'data' может содержать значения "foo" и "bar".

4. Задачи протокола аутентификации

Сервер управляет аутентификацией, говоря клиенту, какой метод аутентификации может использоваться для продолжения обмена в любой момент времени. Клиент волен пробовать предложенные сервером методы в любом порядке. Это предоставляет серверу полный контроль за процессом аутентификации и обеспечивает для клиентов достаточную гибкость выбора наиболее удобного метода из числа поддерживаемых сервером.

Методы аутентификации идентифицируются по их именам, как определено в [SSH-ARCH]. Метод "none" является резервным и его **недопустимо** указывать в числе поддерживаемых. Однако этот метод **может** быть использован клиентом. Сервер **должен** всегда отвергать такие запросы, если клиенту не предоставляется доступ без какой-либо аутентификации (в этом случае сервер **должен** принять такой запрос). Основным назначением запросов является получение от сервера списка поддерживаемых им методов аутентификации.

Серверу **следует** вводить тайм-аут для аутентификации и разрывать соединение, если аутентификация не будет выполнена в течение заданного времени. **Рекомендуемая** продолжительность ожидания составляет 10 минут. Кроме того, реализациям **следует** ограничивать число неудачных попыток аутентификации, которые могут быть предприняты в одной сессии (**рекомендуемое** значение - 20). При превышении заданного числа попыток серверу **следует** разорвать соединение.

Дополнительные рекомендации по времени ожидания и числу попыток аутентификации можно найти в [ssh-1.2.30].

5. Запросы аутентификации

Все запросы на аутентификацию **должны** использовать показанный на врезке формат. Определены только несколько первых полей запроса, а остальные поля зависят от конкретного метода аутентификации.

```
byte      SSH_MSG_USERAUTH_REQUEST
string    имя пользователя в кодировке ISO-10646 UTF-8 [RFC3629]
string    имя службы в кодировке US-ASCII
string    название метода в кодировке US-ASCII
....      поля, используемые данным методом
```

Поля 'user name' и 'service name' повторяются при каждой попытке аутентификации и **могут** менять свои значения. Реализация сервера **должна** аккуратно проверять эти значения в каждом сообщении и сбрасывать все накопленные состояния аутентификации при изменении полей. Если состояние аутентификации невозможно сбросить, соединение **должно** быть разорвано при изменении 'user name' или 'service name'.

Поле 'service name' указывает службу, запускаемую после завершения аутентификации. Может обеспечиваться несколько типов аутентифицированного сервиса. Если запрошенный сервис не поддерживается, сервер **может** разорвать соединение незамедлительно или позднее в любой момент. **Рекомендуется** передавать в таких случаях соответствующее сообщение о разрыве соединения. При любых обстоятельствах, если сервис не существует, для реализации **недопустимо** принимать аутентификацию.

Если пользователя с именем 'user name' не существует, сервер **может** разорвать соединение или передать фиктивный список приемлемых для аутентификации значений 'method name', не принимая вызов ни при каких условиях. Это позволяет серверу предотвратить раскрытие информации о существовании пользователей. При любых обстоятельствах, если 'user name' не существует, **недопустимо** принимать аутентификацию.

Хотя для клиентов обычно нет большого смысла в передаче серверу запросов, которые сервер не включает в число допустимых, передача такого запроса не является ошибкой и серверу **следует** просто отвергнуть запрос, который не удалось распознать.

Запрос аутентификации **может** приводить к дальнейшему обмену сообщениями. Все эти сообщения зависят от используемого метода аутентификации ('method name') и клиент **может** в любой момент передать новое сообщение SSH_MSG_USERAUTH_REQUEST. В ответ на это сервер **должен** отказаться от продолжения предыдущей попытки аутентификации и продолжить обработку новой.

Ниже приведен список определенных значений 'method name'.

```
"publickey"             Обязательно
"password"              Не обязательно
"hostbased"             Не обязательно
"none"                  Не рекомендуется
```

Возможно определение дополнительных значений 'method name' в соответствии с требованиями [SSH-ARCH] и [SSH-NUMBERS].

5.1. Отклики на запросы аутентификации

Если сервер отвергает запрос на аутентификацию, он **должен** передать в ответ сообщение, показанное на врезке.

```
byte        SSH_MSG_USERAUTH_FAILURE
name-list   варианты аутентификации, которые могут быть продолжены
boolean     частичный успех
```

Поле 'authentications that can continue' представляет собой список разделенных запятыми методов аутентификации ('method name'), для которых аутентификационный диалог может продуктивно продолжаться.

Серверам **рекомендуется** включать в список name-list только те имена методов, которые полезны на деле. Однако не запрещено включать в этот список методы, которые не могут быть использованы для аутентификации пользователей.

Успешно выполненные методы аутентификации **не следует** включать в name-list без явной потребности.

Поле 'partial success' (частичный успех) **должно** иметь значение TRUE, если отклик на соответствующий запрос аутентификации был успешным. Если запрос не был успешно обработан, поле должно иметь значение FALSE.

Если сервер принимает аутентификацию, он **должен** передать в ответ сообщение:

```
byte      SSH_MSG_USERAUTH_SUCCESS
```

Отметим, что при использовании множества методов аутентификации это сообщение передается не для каждого метода, а по общему завершению процедуры аутентификации.

Клиент **может** передать несколько аутентификационных запросов, не ожидая отклика на предыдущие запросы. Сервер **должен** полностью обработать каждый запрос полностью и подтвердить неудачный запрос сообщением SSH_MSG_USERAUTH_FAILURE до перехода к обработке следующего запроса.

Запрос, обработка которого требует дополнительного обмена сообщениями, будет прерываться последующим запросом. В таких случаях[1] клиенту **недопустимо** передавать следующий запрос, если он не получил от сервера ответа на предыдущий запрос. Для прерванного метода аутентификации **недопустимо** передавать сообщение SSH_MSG_USERAUTH_FAILURE.

Сообщение SSH_MSG_USERAUTH_SUCCESS **должно** передаваться только один раз. После передачи SSH_MSG_USERAUTH_SUCCESS has все новые запросы аутентификации **следует** отбрасывать без уведомления.

Все не относящиеся к аутентификации запросы клиента, отправленные после запроса, который привел к передаче сообщения SSH_MSG_USERAUTH_SUCCESS, **должны** передаваться службе, работающей «поверх» этого протокола. Такие сообщения идентифицируются по их номерам (см. раздел 6).

5.2. Запрос аутентификации типа "none"

Клиент может запросить список значений 'method name', которые можно «продолжить», используя значение "none" в поле 'method name'.

Если для пользователя не требуется аутентификация, сервер **должен** вернуть сообщение SSH_MSG_USERAUTH_SUCCESS. В остальных случаях сервер **должен** возвращать сообщение SSH_MSG_USERAUTH_FAILURE и **может** вернуть список методов, которые могут быть «продолжены», в поле 'authentications that can continue'.

Это значение[2] 'method name' **недопустимо** указывать в списке методов, поддерживаемых сервером.

5.3. Завершение аутентификации пользователя

Аутентификация завершается, когда сервер ответит сообщением SSH_MSG_USERAUTH_SUCCESS. Все связанные с аутентификацией сообщения, полученные после этого, **следует** отбрасывать без уведомления.

После передачи сообщения SSH_MSG_USERAUTH_SUCCESS сервер запускает запрошенную службу (сервис).

---

[1] В исходном документе этот тезис сформулирован не вполне корректно и кажется противоречащим предыдущему абзацу документа. См. http://www.rfc-editor.org/errata_search.php?rfc=4252&eid=3268. *Прим. перев.*

[2] Значение "none". *Прим. перев.*

5.4. Banner-сообщение

В некоторых странах отправка предупреждений перед аутентификацией может послужить основанием для защиты прав пользователя. По этой причине многие машины UNIX обычно выводят текст из файла /etc/issue, используя TCP wrapper или похожие программы для вывода баннера перед запросом на вход в систему (login prompt).

Сервер SSH может передать сообщение SSH_MSG_USERAUTH_BANNER в любой момент после старта протокола аутентификации и до завершения аутентификации. Это сообщение содержит текст, выводимый пользователю на стороне клиента перед попыткой аутентификации. Формат сообщения показан на врезке.

```
byte      SSH_MSG_USERAUTH_BANNER
string    сообщение в кодировке ISO-10646 UTF-8 [RFC3629]
string    тег языка [RFC3066]
```

По умолчанию клиенту **следует** отобразить сообщение ('message') на экране. Однако, поскольку сообщение явно передается при каждой попытке входа и некоторым клиентам требуется открывать для вывода этого предупреждения новое окно, клиентская программа может позволить пользователю явно запретить вывод баннеров от сервера. Сообщение может содержать множество строк, разделенных парами символов CRLF.

Если строка 'message' отображается на экране, **следует** использовать фильтрацию управляющих символов, описанную в [SSH-ARCH], для предотвращения атак на основе передачи символов управления терминалом.

6. Номера сообщений протокола аутентификации

Все номера сообщений, используемых протоколом аутентификации, лежат в диапазоне от 50 до 79, который является часть пространства номеров, зарезервированного для протоколов, работающих «поверх» транспортного протокола SSH.

Сообщения с номерами 80 и выше зарезервированы для протоколов, работающих после протокола аутентификации, поэтому получение одного из таких сообщений до завершения аутентификации является ошибкой, на которую сервер должен отвечать разрывом соединения (предпочтительно с передачей соответствующего сообщения для упрощения поиска неисправностей).

После успешной аутентификации такие сообщения передаются службам вышележащего уровня.

Ниже перечислены аутентификационные сообщения общего назначения с их номерами.

```
SSH_MSG_USERAUTH_REQUEST            50
SSH_MSG_USERAUTH_FAILURE            51
SSH_MSG_USERAUTH_SUCCESS            52
SSH_MSG_USERAUTH_BANNER             53
```

В дополнение к сказанному выше следует отметить, что диапазон номеров от 60 до 79 зарезервирован для сообщений, зависящих от метода. Такие сообщения передаются только сервером (клиент передает только сообщения SSH_MSG_USERAUTH_REQUEST). Различные методы аутентификации могут использовать одинаковые номера сообщений из указанного диапазона.

7. Метод аутентификации с открытым ключом - "publickey"

Единственным **требуемым** методом аутентификации ('method name') является "publickey" (аутентификация по открытому ключу). Все реализации **должны** поддерживать этот метод, однако открытые ключи есть не у каждого пользователя и локальное законодательство большинства стран не требует аутентификации по открытым ключам для всех пользователей в настоящее время и на ближайшее будущее.

В этом методе для аутентификации пользователя служит наличие секретного ключа. Метод реализуется путем передачи цифровой подписи (сигнатуры), создаваемой на основе секретного ключа пользователя. Сервер **должен** удостовериться, что ключ является корректным идентификатором для пользователя, а также **должен** проверить корректность подписи. При положительном результате обеих проверок запрос **должен** быть воспринят, а в противном случае запрос **должен** быть отвергнут. Отметим, что сервер **может** потребовать дополнительной аутентификации после успешной проверке открытого ключа.

Секретные ключи зачастую хранятся в зашифрованном виде на хосте клиента и пользователь должен ввести пароль (passphrase), чтобы была сгенерирована цифровая подпись. Даже если это не так, генерирование цифровой подписи требует достаточно интенсивных расчетов. Для предотвращения избыточной обработки с участием пользователя используется показанное на врезке справа сообщение, служащее для запроса информации о подходящем методе аутентификации с использованием открытого ключа ("publickey").

```
byte      SSH_MSG_USERAUTH_REQUEST
string    имя пользователя в кодировке ISO-10646 UTF-8 [RFC3629]
string    имя службы в кодировке US-ASCII
string    "publickey"
boolean   FALSE
string    имя алгоритма открытого ключа
string    открытый ключ (blob)
```

Алгоритмы открытых ключей определены в спецификации транспортного уровня [SSH-TRANS]. Поле 'public key blob' (открытый ключ) может содержать сертификаты.

При аутентификации может быть предложен любой из алгоритмов открытых ключей. В частности, список не ограничивается тем, что было согласовано в процессе обмена ключами. Если сервер не поддерживает тот или иной алгоритм, он **должен** просто отвергнуть запрос.

```
byte      SSH_MSG_USERAUTH_PK_OK
string    имя алгоритма открытого ключа из запроса
string    открытый ключ (blob) из запроса
```

Сервер **должен** ответить на это сообщение своим сообщением SSH_MSG_USERAUTH_FAILURE или сообщением, показанным на врезке слева.

Для выполнения реальной аутентификации клиент тогда **может** отправить подпись, сгенерированную с использованием секретного ключа (private key). Клиент **может** передать подпись напрямую без предварительной проверки приемлемости ключа. Формат подписи показан на врезке справа.

```
byte      SSH_MSG_USERAUTH_REQUEST
string    имя пользователя
string    имя службы
string    "publickey"
boolean   TRUE
string    имя алгоритма открытого ключа
string    открытый ключ для использования при аутентификации
string    подпись
```

```
string    идентификатор сессии
byte      SSH_MSG_USERAUTH_REQUEST
string    имя пользователя
string    имя службы
string    "publickey"
boolean   TRUE
string    имя алгоритма открытого ключа
string    открытый ключ для использования при аутентификации
```

Значением поля 'подпись' ('signature') является сигнатура приведенных на врезке слева данных (с сохранением их порядка), полученная с помощью соответствующего секретного ключа.

Когда сервер получает такое сообщение, он **должен** проверить, подходит ли предъявленный ключ для аутентификации, и (при положительном результате) **должен** проверить корректность цифровой подписи (сигнатуры).

При положительном результате обеих проверок аутентификация считается успешной. Отметим, что сервер **может** потребовать дополнительной аутентификации. По результатам аутентификации сервер **должен** передать сообщение SSH_MSG_USERAUTH_SUCCESS (если дополнительной аутентификации не требуется) или SSH_MSG_USERAUTH_FAILURE (если обработка запроса завершилась отказом или требуется дополнительная аутентификация).

Для метода аутентификации "publickey" определено зависящее от метода сообщение

```
SSH_MSG_USERAUTH_PK_OK              60
```

8. Метод парольной аутентификации - "password"

Для парольной аутентификации используется сообщение, показанное на врезке справа. Отметим, что сервер **может** запросить у пользователя смену пароля. Всем реализациям **следует** поддерживать парольную аутентификацию.

```
byte      SSH_MSG_USERAUTH_REQUEST
string    имя пользователя
string    имя службы
string    "password"
boolean   FALSE
string    нешифрованный пароль в кодировке ISO-10646 UTF-8 [RFC3629]
```

Отметим, что нешифрованный пароль ('plaintext password') передается в кодировке ISO-10646 UTF-8. Интерпретация пароля и сравнение полученного значения с базой данных о паролях сервер определяет самостоятельно. Однако, если клиент считывает пароль в другой кодировке (например, ISO 8859-1 – ISO Latin1), он **должен** преобразовать пароль в кодировку ISO-10646 UTF-8 до передачи, а сервер **должен** преобразовать полученный пароль в кодировку, используемую его системой для паролей.

С точки зрения поддержки разных языков желательно, чтобы процесс парольной аутентификации работал независимо от операционной системы и применяемой пользователем клиентской программы. Для этого требуется нормализация. Системам, поддерживающим пароли в кодировках, отличных от ASCII, **следует** всегда нормализовать пароли и имена пользователей при их включении в базу данных и сравнении (с хэшированием или без него) с имеющимися в базе данных записями. Реализациям SSH, которые сохраняют и сравнивают пароли, **следует** использовать нормализацию в соответствии с [RFC4013].

Отметим, что передаваемый в открытом виде пароль шифруется вместе с пакетом в целом на транспортном уровне. Как серверу, так и клиенту следует проверять обеспечение конфиденциальности нижележащим транспортным уровнем (т. е., использование шифрования). Если конфиденциальность на транспортном уровне не обеспечивается (шифрование "none"), парольную аутентификацию **следует** отключить. Если не обеспечивается конфиденциальность и MAC, изменение пароля **следует** запретить.

Обычно сервер отвечает на эти сообщения откликом об успехе или отказе. Однако при окончании срока действия пароля серверу **следует** указать на это, отвечая сообщением SSH_MSG_USERAUTH_PASSWD_CHANGEREQ (см. врезку). В любом случае серверу **недопустимо** принимать для аутентификации просроченный пароль.

```
byte      SSH_MSG_USERAUTH_PASSWD_CHANGEREQ
string    приглашение в кодировке ISO-10646 UTF-8 [RFC3629]
string    тег языка [RFC3066]
```

```
byte      SSH_MSG_USERAUTH_REQUEST
string    имя пользователя
string    имя службы
string    "password"
boolean   TRUE
string    нешифрованный старый пароль в кодировке ISO-10646 UTF-8 [RFC3629]
string    нешифрованный новый пароль в кодировке ISO-10646 UTF-8 [RFC3629]
```

При получении приглашения сменить пароль клиент **может** перейти к другому методу или запросить у пользователя новый пароль и повторить парольную аутентификацию, используя показанное на врезке слева сообщение. Клиент также **может** использовать это сообщение вместо обычного запроса парольной аутентификации без запроса на смену пароля со стороны сервера.

Сервер должен отвечать на каждый запрос сообщением SSH_MSG_USERAUTH_SUCCESS, SSH_MSG_USERAUTH_FAILURE или другим запросом SSH_MSG_USERAUTH_PASSWD_CHANGEREQ. Смысл этих сообщений раскрыт ниже.

SSH_MSG_USERAUTH_SUCCESS - пароль был изменен, аутентификация успешно завершена.

SSH_MSG_USERAUTH_FAILURE с частичным успехом (partial success) - пароль был изменен, но требуется дополнительная аутентификация.

SSH_MSG_USERAUTH_FAILURE без частичного успеха - пароль не был изменен (смена пароля не поддерживается или старый пароль указан некорректно). Отметим, что получение от сервера запроса SSH_MSG_USERAUTH_PASSWD_CHANGEREQ говорит о поддержке возможности смены пароля.

SSH_MSG_USERAUTH_CHANGEREQ - пароль не был изменен, поскольку новый пароль оказался неприемлемым (например, слишком простой).

Для метода парольной аутентификации определено зависящее от метода сообщение

```
SSH_MSG_USERAUTH_PASSWD_CHANGEREQ    60
```

9. Аутентификация по хосту - "hostbased"

Для некоторых сайтов может быть желательна аутентификация по хосту и имени пользователя на удаленном хосте. Хотя такой метод аутентификации неприемлем для сайтов с высоким уровнем требований по безопасности, он может оказаться весьма удобным во многих случаях. Эта форма аутентификации является **необязательной**. При использовании такой аутентификации **следует** принимать меры против доступа рядовых пользователей к секретному ключу хоста.

Клиент запрашивает аутентификацию по этому методу путем передачи показанного ниже сообщения. Это похоже на используемые в UNIX варианты аутентификации "rhosts" и "hosts.equiv", за исключением того, что клиентских хост проверяется более строго.

Этот метод работает на основе передачи клиентом цифровой подписи (сигнатуры), созданной с использованием секретного ключа клиентского хоста, которую сервер проверяет, используя открытый ключ этого хоста. После проверки клиентского хоста выполняется проверка полномочий пользователя (но не дополнительная аутентификация) по его имени на сервере и клиенте, а также имени клиентского хоста.

```
byte      SSH_MSG_USERAUTH_REQUEST
string    имя пользователя
string    имя службы
string    "hostbased"
string    алгоритм открытого ключа для ключа хоста
string    открытый ключ хоста и сертификаты для клиентского хоста
string    имя клиентского хоста в формате FQDN и кодировке US-ASCII
string    имя пользователя на клиентском хосте в кодировке in ISO-10646 UTF-8 [RFC3629]
string    подпись
```

Имена алгоритмов открытых ключей ('public key algorithm for host key') определены в спецификации транспортного уровня [SSH-TRANS]. Поле 'public host key and certificates for client host' может включать сертификаты.

Поле 'signature' содержит цифровую подпись, созданную с использованием секретного ключа хоста, для приведенных ниже данных (с сохранением их порядка).

```
string    идентификатор сессии
byte      SSH_MSG_USERAUTH_REQUEST
string    имя пользователя
string    имя службы
string    "hostbased"
string    алгоритм открытого ключа для ключа хоста
string    открытый ключ хоста и сертификаты для клиентского хоста
string    имя клиентского хоста в формате FQDN и кодировке US-ASCII
string    имя пользователя на клиентском хосте в кодировке in ISO-10646 UTF-8 [RFC3629]
```

Сервер **должен** убедиться, что ключ хоста действительно относится к указанному в сообщении клиентскому хосту, данному пользователю разрешен вход на данный хост, а значение 'signature' является корректной цифровой подписью, созданной с данным ключом хоста. Сервер **может** игнорировать имя пользователя ('user name'), если он считает нужным аутентифицировать только хост клиента.

При наличии возможности **рекомендуется** выполнять на сервере дополнительные проверки, позволяющие убедиться, что адрес полученный из (недоверенной) сети, соответствует данному имени клиентского хоста. Такая проверка осложнит использование скомпрометированных хостов. Отметим, что при этом может потребоваться специальная обработка для соединений, проходящих через межсетевые экраны.

10. Согласование с IANA

Этот документ является частью комплекта документов. Вопросы согласования с агентством IANA для протокола SSH, определенного в [SSH-ARCH], [SSH-TRANS], [SSH-CONNECT] и данном документе, детализированы в [SSH-NUMBERS].

11. Вопросы безопасности

Целью этого протокола является выполнение аутентификации пользователей. Предполагается, что протокол работает на базе защищенного протокола транспортного уровня, который уже аутентифицирован серверной машиной, организовал шифрованный коммуникационный канал и рассчитал уникальный идентификатор для данной сессии. Транспортный уровень обеспечивает защиту конфиденциальности для парольной аутентификации и других методах, основанных на секретной информации.

Полное рассмотрение вопросов безопасности для этого протокола приведено в [SSH-ARCH].

12. Литература

12.1. Нормативные документы

[SSH-ARCH] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Protocol Architecture", RFC 4251[1], January 2006.

[SSH-CONNECT] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Connection Protocol", RFC 4254[1], January 2006.

[SSH-TRANS] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253[1], January 2006.

[SSH-NUMBERS] Lehtinen, S. and C. Lonvick, Ed., "The Secure Shell (SSH) Protocol Assigned Numbers", RFC 4250[1], January 2006.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119[1], March 1997.

[RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434[1], October 1998.

[RFC3066] Alvestrand, H., "Tags for the Identification of Languages", BCP 47, RFC 3066, January 2001.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC4013] Zeilenga, K., "SASLprep: Stringprep Profile for User Names and Passwords", RFC 4013, February 2005.

12.2. Дополнительная литература

[ssh-1.2.30] Ylonen, T., "ssh-1.2.30/RFC", файл из архива ftp://ftp.funet.fi/pub/unix/security/login/ssh/ssh-1.2.30.tar.gz, November 1995.

Адреса авторов

Tatu Ylonen

SSH Communications Security Corp

Valimotie 17

00380 Helsinki

Finland

EMail: ylo@ssh.com

Chris Lonvick (htlfrnjh)

Cisco Systems, Inc.

12515 Research Blvd.

Austin 78759

USA

EMail: clonvick@cisco.com

Перевод на русский язык

Николай Малых

nmalykh@gmail.com

Торговые марки

ssh – торговый знак, зарегистрированный в США и/или других странах.

[1]Перевод этого документа на русский язык доступен на сайте http://www.protocols.ru. *Прим. перев.*

under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Подтверждение

Финансирование функций RFC Editor обеспечено IETF Administrative Support Activity (IASA).